Git

Introduction

Système de contrôle de version (VCS)

- sauvegarder son code source
- mémoriser et retrouver différentes version d'un projet
- faciliter le travail collaboratif (interfaces web github/gitlab)
- logiciel libre / opensource
- le plus populaire
- s'installe à peu près partout

Pourquoi ?

- sauvegarder des états du code source
  - > états intermédiaires de développement :
    - 1 fonctionnalité,
    - 1 correction de bug,
    - fin de la journée (!)
  - > éviter de tout perdre : mauvaise manip.
  - > pouvoir revenir en arrière
  - > versions distribuées aux utilisateurs
- éviter des copies “bêtes” $\rightarrow$ prend de la place inutilement
- gérer plusieurs états de devéloppement en parallèle cf. notion de *branche*

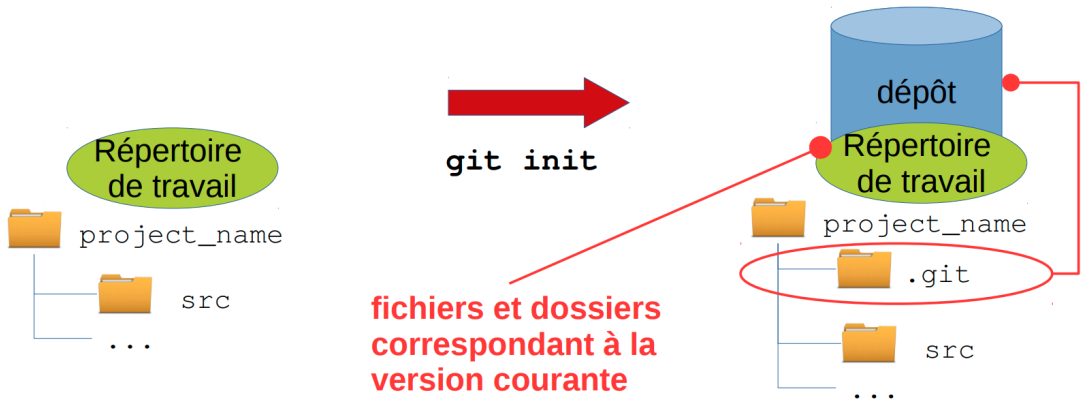
Comment ?

- sauvegarde un état initial
- ensuite ne sauvegarde que les différences ligne par ligne
  - > 1 sauvegarde $=$ 1 **commit**
- adapté au code source de type texte, pas binaire (git-lfs)
- système décentralisé
  - > copie locale Vs. copie de référence sur un serveur (remote)
  - > copie locale peut être $\neq$ image serveur
  - > copie locale : développer dans votre coin sans gêner les autres, sans être polué par les autres
  - > image serveur : état de référence pour tout le monde propre/testé $\rightarrow$ github/gitlab

Inria

1. créer son dépôt à partir d'un répertoire

```
cd project/
git init
git add ...
git commit ...
git remote add origin git@gitlab.com:group/project.git
git push
```

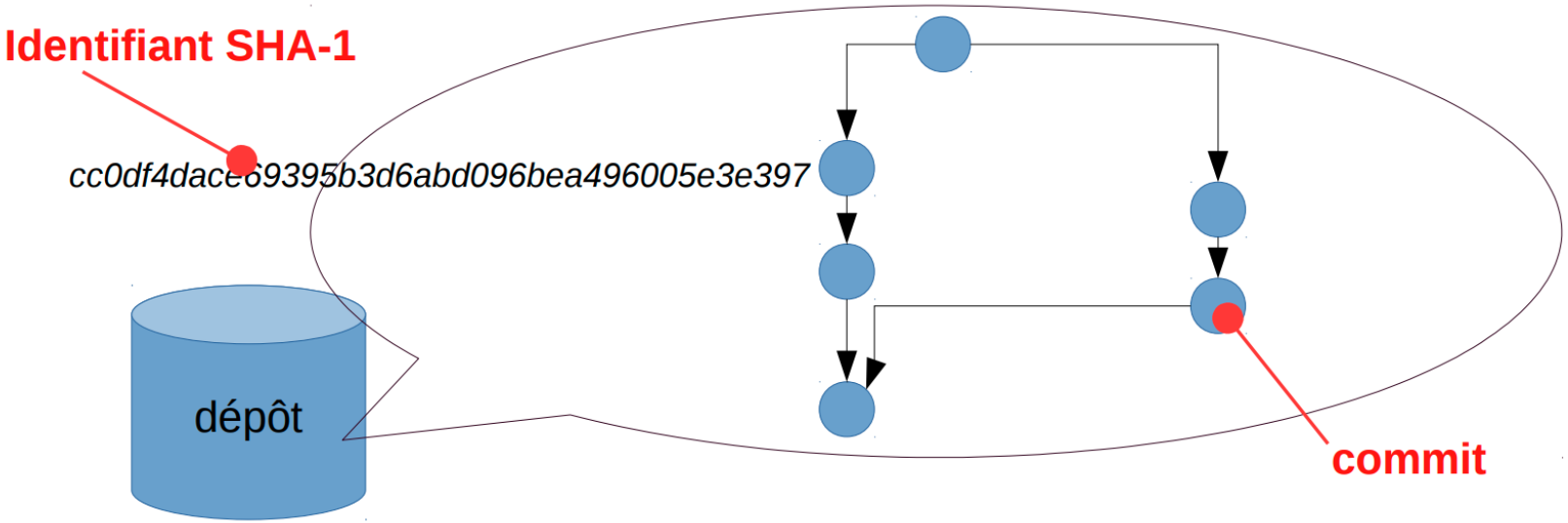


2. "cloner" à partir d'un dépôt existant

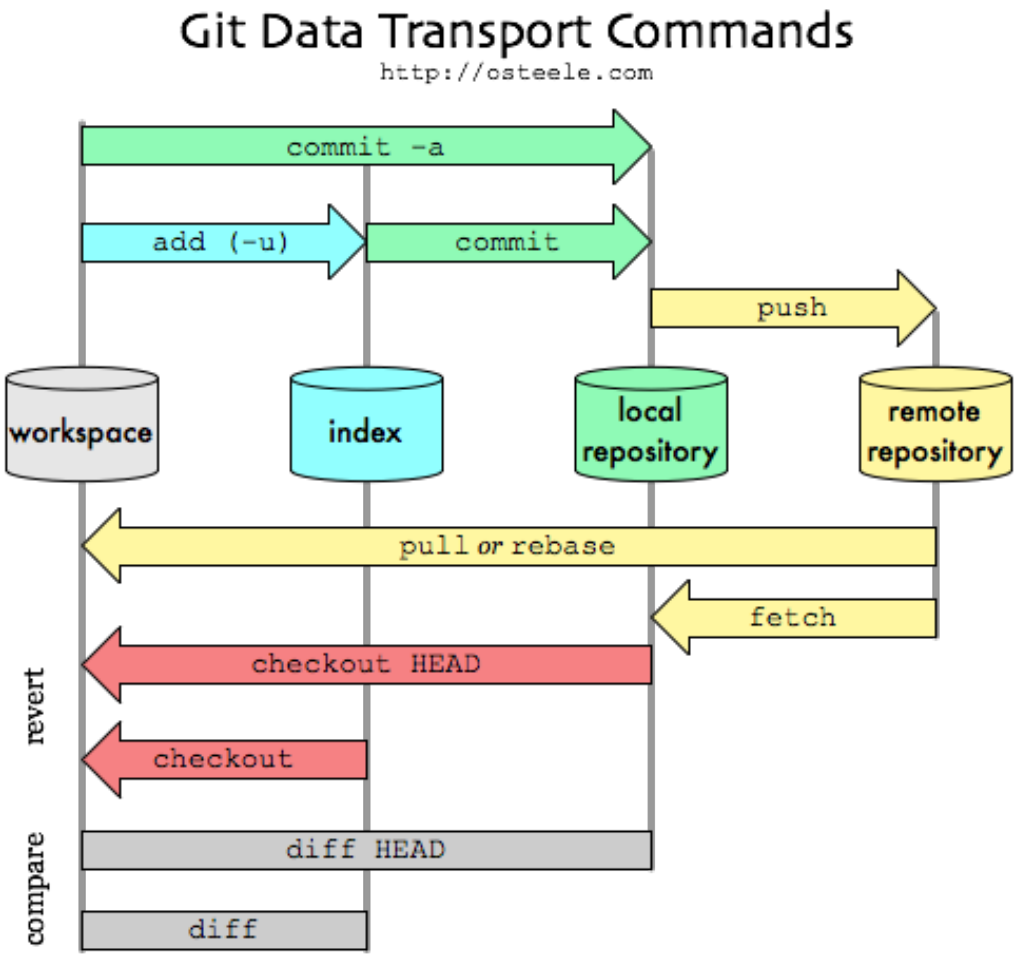
```
git clone git@gitlab.com:group/project.git
```

Commit

- dépôt git = graphe (orienté acyclique) de commits
- commit = ensemble de modifications dans des fichiers
- identifié avec un hash code



Git area


Git Data Transport Commands
http://osteele.com
commit -a
add (-u)
commit
push
workspace
index
local repository
remote repository
pull or rebase
fetch
checkout HEAD
revert
checkout
compare
diff HEAD
diff

Petit tour des commandes/opérations

- **git add [-p] [-u]** : ajout de fichiers/modifications dans l'index
- **git rm fichiers** : enlever des fichiers de l'index
- **git commit fichiers -m "message"** : sauvegarde modifs.
- **git branch b** : création de la branche "b"
- **git branch -d b** : suppression de branche
- **git checkout b** : se positionne sur branche "b"
- **git checkout f.txt** : annule les modifications en cours
- **git merge/rebase a** : fusion entre notre branche et "a"
- **git push** : pousse les commits sur le serveur
- **git fetch** : récupération de l'état serveur dans l'index
- **git pull** : récupère les commits du serveur (=fetch+merge)
- **git reset [–soft] [–hard]** <**commit**> : annuler des commits
- **git diff/status/log** : état répertoire/graphe de commits

Inria

- Fichier .gitconfig : configuration générale

```
git config --global user.name "John Doe"
git config --global user.email doe@youpi.fr
git config --global color.ui auto
git config --global core.editor "code --wait"
git config --global alias.ci commit
git config --global alias.glog log --all --graph \
           --oneline --decorate=short \
           --branches='*' --tags='*'

cat ~/.gitconfig
```

- Fichier .gitignore : fichiers à exclure
  - local au projet, ex. mettre *.class
  - ou fichier personnel global

```
git config --global core.excludesfile ~/.gitignore_global
```

- Forker un projet existant: `https://gitlab-ce.iut.u-bordeaux.fr:Pierre/DEMO-GIT-PT2`
- Ajout clef ssh dans Gitlab
- Cloner votre projet
- Compiler le projet java
- Prise en main du code
  - installer vscode
  - extensions git, java
  - exécuter le programme *Life*

Etat des sources

- Modifier le code source Life.java
- git status
- git diff
- voir dans vscode

Annuler les modifs. en cours

- git checkout fichiers
- git checkout .
- idem git restore
- voir dans vscode

Sauvegarde avec un commit

- git commit fichiers -m “message”

ou

- git add fichiers
- git commit -m “message”

Se positionner sur un commit

- git checkout <commit>
- git chekout master

Branches

- git branch bricole
- git branch [-a]
- git checkout bricole
- git switch master
- git branch -d/-D bricole
- git switch -c bricole
- git status

Supprimer les derniers commits

- git reset [- -hard]

Bien préparer son commit

- git add [-p] [-u]
- git restore - -staged (ou git reset)
- git commit -m “message”

Fusion de branche avec merge

- sur master modif. Life.java L.30, commit
- git sw bricole
- modif. Life.java L.30, commit
- git sw master
- git merge bricole $\rightarrow$ conflit
- on édite Life.java, on résout le conflit
- git add
- git merge - -continue, message de fusion
- git merge - -abort, pour abandonner fusion

Inria

Fusion de branche avec rebase

- sur master modif. Life.java L.30, commit
- git sw bricole
- modif. Life.java L.31, commit
- git rebase master
- git log, on a bien le commit de master puis celui de bricole

Pousser sa branche sur le dépôt

- git push -u origin bricole, cf. vue gitlab
- git push origin - -delete bricole

Petit retour en arrière avant la suite

- git sw master
- git br -D bricole
- git reset - -hard HEAD~1

Récupérer les commits du serveur

- cloner le projet dans /tmp
- commit d'une modif
- git push

dans sa 1re copie locale du projet faire un commit au même endroit

- git fetch
- git diff origin/master $\rightarrow$ un pull va faire un conflit
- git reset - -hard HEAD$\sim$1, git pull (ok)

A faire que si la branche n'a pas déjà été clonnée par d'autres

- git reset - -hard HEAD~1
- git push - -force

rejet car branche master est protégée côté serveur (gitlab)

- "unprotect branch" dans gitlab (settings, repository)
- git push - -force
- restaurer la protection

- git checkout - -track origin/bricole

ou, en deux temps

- git branch bricole - -track origin/bricole
- git sw bricole

ou (plus moderne)

- git switch -c test -t origin/test

si branches distances supprimées, faire le ménage

- git remote prune origin

Mettre de côté des modifs. pour plus tard

- modif. fichier
- git stash
- git status
- git stash pop

pour supprimer une modif. sans l'appliquer

- git stash drop

Ajout d'un serveur distant "remote"

par exemple pour suivre le projet forké d'origine

- git remote add upstream git@gitlab-ce.iut.u-bordeaux.fr:Pierre/DEMO-GIT-PT2.git
- git remote show [origin]
- git remote show [upstream]

récupérer dernier état d'un autre serveur distant

- git pull upstream master

Inria